AF
JFW

Docket No.: 1075.1174

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Akio MATSUDA, et al.

Serial No. 09/964,591 Group Art Unit: 2123

Confirmation No. 2406

Filed: September 28, 2001 Examiner: Kandasamy THANGAVELU

For: METHOD OF SIMULATING OPERATION OF LOGICAL UNIT, AND COMPUTER-READABLE RECORDING MEDIUM RETAINING PROGRAM FOR SIMULATING OPERATION OF LOGICAL UNIT

COMMUNICATION IN RESPONSE TO NOTIFICATION OF NON-COMPLIANT APPEAL BRIEF

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

Attached is a Substitute Appeal Brief in response to the Notification of Non-Compliant Appeal Brief mailed September 7, 2007. In the Notification, box 2 is checked indicating that the brief does not identify the appealed claims. In the Notification, below block 10 is indicated "Section III Status of claims section does not indicate claims on appeal." (See, block 10 note).

In the attached Substitute Appeal Brief, Section III identifies all pending claims 1-41 as on appeal.

If there are any additional fees associated with filing of this Substitute Appeal Brief, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: October 5, 2007

By: Paul W Bobowiec
Paul W. Bobowiec
Registration No. 47,431

1201 New York Ave, N.W., 7th Floor
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

Docket No. 1075.1174

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Akio MATSUDA, et al.

Serial No. 09/964,591 Group Art Unit: 2123

Confirmation No. 2406

Filed: September 28, 2001 Examiner: Kandasamy THANGAVELU

For: METHOD OF SIMULATING OPERATION OF LOGICAL UNIT, AND COMPUTER-READABLE RECORDING MEDIUM RETAINING PROGRAM FOR SIMULATING OPERATION OF LOGICAL UNIT

SUBSTITUTE APPEAL BRIEF

Mail Stop Appeal Brief-Patents
Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Sir:

In response to a Notification of Non-Compliant Appeal Brief mailed on September 7, 2007, a Substitute Appeal Brief is submitted herewith.

I. REAL PARTY IN INTEREST

The real party in interest is Fujitsu Limited, the assignee of the subject application.

II. RELATED APPEALS AND INTERFERENCES

Appellants, Appellants' legal representatives, and assignee are not aware of any prior or pending appeals or interferences which directly affect or are directly affected by, or have a bearing, on the Board's decision in the pending appeal.

III. STATUS OF CLAIMS

Claims 1-41 are pending, stand rejected under 35 U.S.C. §103(a), and are on appeal.

Claims 1, 3, 6, 13-14, 17, 32 and 35 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan (U.S.P. 6,466,898) (Chan) in view of Dearth et al. (U.S.P. 6,345,242) (Dearth '242), Dearth et al. (U.S.P. 5,812,824) (Dearth '824),and Levy et al. (U.S.P. 6,092,175) (Levy) and are on appeal.

(Appellants respectfully points out that throughout the Office Action, the Examiner incorrectly refers to Chan as "Chen.")

Claims 2, 23, and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824, Levy, and further in view of Kinzelman et al. (U.S.P. 5,594,741) (Kinzelman) and are on appeal.

Claims 4-5, 15-16, and 33-34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, Levy, and further in view of De Yong et al. (U.S.P. 5,355,435) (De Yong) and are on appeal.

Claims 7, 18, and 36 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Levy and are on appeal.

Claims 8, 10, 19, 21, 37, and 39 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Markov (U.S. P. 6,314,552) (Markov) and are on appeal.

Claims 9, 20, and 38 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Markov and Kasuya (U.S.P. 6,077,304) Kasuya and are on appeal.

Claims 11, 22, and 40 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Furuichi (U.S. P. 5,437,037) Furuichi and are on appeal.

Claims 12 and 41 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Hollander (U.S.P. 6,347,388) and are on appeal..

Claims 24 and 25 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of

Kinzelman and De Yong and are on appeal.

Claim 27 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman and is on appeal..

Claims 28 and 30 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Kinzelman and Markov and are on appeal..

Claim 29 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Kasuya (U.S.P. 6,077,304) Kasuya and is on appeal.

Claim 31 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Furuichi and is on appeal.

IV. STATUS OF AMENDMENTS

No amendment has been filed subsequent to the rejection made in the Office Action mailed January 22, 2007.

V. SUMMARY OF CLAIMED SUBJECT MATTER

Paragraph numbers refer to paragraph numbers of the substitute specification filed May 10, 2005.

The claimed invention in independent claim 1 recites a method of simulating an operation of a logical unit including requesting a resource in which a thread manager controls threads each forming an execution unit of a program (see, for example, Fig. 2, thread manager 11, threads 13 and paragraphs [0047] and [0050]). The method of claim 1 also includes the thread manager making a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of a series of functions required until the operation of the logical unit reaches completion according to a design specification of the logical unit (see, for example, Fig. 2, thread manager 11, resource 14, e.g., resource data, Fig. 5, processing, and Fig. 8, step 21, and paragraphs [0050]-[0052], [0081], and [0118]).

The method of claim 1 also includes the thread manager making the request to a resource manager which manages the information about the hardware resource (see, for example, Fig. 2, resource manager 14, resource 14, e.g., resource data, and paragraph [0055].) The method of claim 1 also includes allocating a resource in which the resource manager allocates the hardware resource meeting the request to the thread in accordance with a rule prescribed in advance (see, for example, Fig. 2, resource manager 12, resource 14, e.g., resource data, Fig. 3, resource manager 12, and Fig. 6, arbitration rule, and paragraphs [0054], [0066], and [0072].)

The method of claim 1 also includes controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager (see, for example, Fig. 3, thread manager 11, thread 13, resource manager 12, and paragraphs [0064] -[0068]).

The method of claim 1 also includes the thread manager and the resource manager executing the requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of the thread reaches completion. (see, for example, paragraph [0069]).

The method of claim 1 also includes the thread manager and the resource manager executing the requesting, allocating, and controlling while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed. (see, for example, Fig. 8, step 24, and paragraphs [0073] and [0089]). The method of claim 1 also includes simulating the

operation of the logical unit to be conducted up to the completion. (see, for example, Fig. 4, and paragraph [0070]).

The claimed invention in independent claim 12 recites a method of simulating an operation of a logical unit including requesting a resource in which a thread manager controls threads each forming an execution unit of a program. (see, for example, Fig. 2, thread manager 11, threads 13 and paragraphs [0047] and [0050]. The method of claim 12 also includes the thread manager making a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of functions required until the operation of the logical unit reaches completion according to a design specification of the logical unit. (see, for example, Fig. 2, thread manager 11, resource 14, e.g., resource data, and paragraphs [0050]-[0052].)

The method of claim 12 includes allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance. (see, for example, Fig. 2, resource manager 12, resource 14, e.g., resource data, Fig. 3, resource manager 12, and Fig. 6, arbitration rule, and paragraphs [0054], [0066], and [0072].)

The method of claim 12 includes controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager. (see, for example, Fig. 3, thread manager 11, thread 13, resource manager 12, and paragraphs [0064] -[0068]).

The method of claim 12 includes with the thread manager and the resource manager executing the requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of the thread reaches completion. (see, for example, paragraph [0069]).

The method of claim 12 includes with the thread manager and the resource manager executing the requesting, allocating, and controlling while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed, simulating the operation of the logical unit to be conducted up to the completion. (see, for example, Fig. 8, step 24, and paragraphs [0073] and [0089]).

The method of claim 12 includes comparing a result of the simulation with an estimated value on the operation of the logical unit (see, for example, Fig. 8, step s27, and paragraphs [0094]-[0095]). The method of claim 12 also includes outputting a result of the comparison to an external unit. (see, for example, Fig. 1, display 3, and paragraphs [0041] and [0098]).

The claimed invention in independent claim 13 recites an apparatus for simulating an operation of a logical unit including a thread manager for controlling a thread forming an execution unit of a program. The apparatus of claim 13 includes a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of the thread.

The apparatus of claim 13 further includes the thread manager including resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread representative of functions required until the operation of the logical unit reaches completion according to a design specification of the logical unit, to the resource manager. (see, for example, Fig. 2, thread manager 11, resource 14, e.g., resource data, and paragraphs [0050]-[0052].)

The apparatus of claim 13 further includes the thread manager including thread control means for controlling an execution state of the thread in accordance with a result of a resource allocation made by the resource manager in response to the request from the resource requesting means. (see, for example, Fig. 3, thread manager 11, thread 13, resource manager 12, and paragraphs [0064] -[0068]). The apparatus of claim 13 further includes the resource manager including resource allocating means for allocating information about a hardware resource relating to a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance. (see, for example, Fig. 2, resource manager 12, resource 14, e.g., resource data, Fig. 3, resource manager 12, and Fig. 6, arbitration rule, and paragraphs [0054], [0066], and [0072].)

The apparatus of claim 13 includes the thread manager and the resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other until the execution of the thread reaches completion(see, for example, paragraph [0069]). The apparatus of claim 13 includes the thread manager and the resource manager conducting the resource request and the control of the thread execution state while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed, for simulating the operation of the logical unit to be conducted up to the completion. (see, for example, Fig. 8, step 24, and paragraphs [0073] and [0089]).

The claimed invention in independent claim 14 recites a computer-readable recording medium storing a program for simulation of an operation of a logical unit of claim 14 includes a program comprising computer instructions which when executed on a computer makes the

computer function as a thread manager for controlling threads each forming an execution unit of the program and as a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of each of threads, by requesting a resource in which the thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of threads representative of a series of functions required until the operation of the logical unit reaches completion according to a design specification of the logical unit, to the resource manager. (see, for example, Fig. 2, thread manager 11, resource 14, e.g., resource data, and paragraphs [0050]-[0052].)

The program of claim 14 further allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance. (see, for example, Fig. 2, resource manager 12, resource 14, e.g., resource data, Fig. 3, resource manager 12, and Fig. 6, arbitration rule, and paragraphs [0054], [0066], and [0072].)

The program of claim 14 further controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager. (see, for example, Fig. 3, thread manager 11, thread 13, resource manager 12, and paragraphs [0064] -[0068]). The program of claim 14 including the thread manager and the resource manager executing the requesting, the allocating, and the controlling repeatedly in cooperation with each other until the execution of the thread reaches completion. (see, for example, paragraph [0069]).

The program of claim 14 including said thread manager and said resource manager executing the requesting, the allocating, and the controlling while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed, simulating the operation of the logical unit to be conducted up to the completion. (see, for example, Fig. 8, step 24, and paragraphs [0073] and [0089]).

The claimed invention in independent claim 41 recites a computer-readable recording medium storing a computer-readable program for simulation of an operation of a logical unit, the program comprising computer instructions which when executed on a computer makes the computer execute requesting a resource in which a thread manager, which controls threads each forming an execution unit of a program, makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of functions required until the operation of the logical unit reaches completion according to a

design specification of the logical unit, to a resource manager which manages the information about a hardware resource. (see, for example, Fig. 2, thread manager 11, resource 14, e.g., resource data, and paragraphs [0050]-[0052].)

Claim 41 further includes a program comprising computer instructions which when executed on a computer makes the computer execute allocating a resource in which the resource manager allocates the hardware resource meeting the request to the thread in accordance with a rule prescribed in advance. (see, for example, Fig. 2, resource manager 12, resource 14, e.g., resource data, Fig. 3, resource manager 12, and Fig. 6, arbitration rule, and paragraphs [0054], [0066], and [0072].)

Claim 41 further includes a program the resource manager controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager. (see, for example, Fig. 3, thread manager 11, thread 13, resource manager 12, and paragraphs [0064] -[0068]). Claim 41 further includes a program including the thread manager and the resource manager executing the requesting, the allocating, and the controlling repeatedly in cooperation with each other until the execution of the thread reaches completion. (see, for example, paragraph [0069]).

Claim 41 further includes a program including the thread manager and the resource manager executing the requesting, the allocating, and the controlling while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed, simulating the operation of the logical unit to be conducted up to the completion. (see, for example, Fig. 8, step 24, and paragraphs [0073] and [0089]).

Claim 41 further includes a program comprising computer instructions which when executed on a computer makes the computer execute comparing a result of the simulation with an estimated value on the operation of the logical unit (see, for example, Fig. 8, step s27, and paragraphs [0094]-[0095]). Claim 41 further includes outputting a result of the comparison to an external unit. (see, for example, Fig. 1, display 3, and paragraph [0041]).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

In the Office Action, claims 1, 3, 6, 13-14, 17, 32 and 35 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824, and Levy.

Claims 2, 23, and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824, Levy, and further in view of Kinzelman.

Claims 4-5, 15-16, and 33-34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, Levy, and further in view of De Yong.

Claims 7, 18, and 36 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Levy.

Claims 8, 10, 19, 21, 37, and 39 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Markov.

Claims 9, 20, and 38 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Markov and Kasuya.

Claims 11, 22, and 40 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Furuichi.

Claims 12 and 41 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Hollander.

Claims 24 and 25 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman and De Yong.

Claim 27 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman.

Claims 28 and 30 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Kinzelman and Markov.

Claim 29 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in

view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Kasuya (U.S.P. 6,077,304) Kasuya.

Claim 31 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Furuichi.

Claims are each independently patentable over the references as set forth below, and do not stand or fall together.

VII. ARGUMENT

All arguments are directed to the grounds of rejection. All citations to the "Office Action" refer to the last Office Action of January 22, 2007. (Appellants respectfully points out that throughout the Office Action, the Examiner incorrectly refers to Chan as "Chen.")

A. The Law Regarding the Obviousness Issues

To establish obviousness under §103, the Examiner must consider the claimed invention "as a whole," and the prior art must teach or suggest all of the claim features. *See* Manual Of Patent Examining Procedure § 2143.03 (8th ed. Rev. 2 May 2004)("MPEP"); *In re Royka*, 180 U.S.P.Q. 580, 583 (C.C.P.A. 1974); *In re Fine*, 5 U.S.P.Q. 2d 1596, 1599 (Fed. Cir. 1988); *Ruiz v. A.B. Chance Co.*, 69 U.S.P.Q.2d 1686, 1690 (Fed. Cir. 2004). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970).

If an independent claim is nonobvious under 35 U.S.C. §103, then any claim depending therefrom is nonobvious. *In re Fine*, 837 F.2d 1071, 5 USPQ 2d 1596 (Fed. Cir. 1988). See Manual Of Patent Examining Procedure § 2143.03 (8th ed. Rev. 5 August 2006)("MPEP").

In *KSR International Co. v. Teleflex* Inc., 82 USPQ2d 1385, 127 SCt 1727, 167 LEd2d 705 (U.S. 2007), the U.S. Supreme Court held that in determining obviousness, one "must ask whether the improvement is more than the predictable use of prior art elements according to their established functions" slip op. 13, 82 USPQ2d at 1396. Furthermore, it is necessary "to determine whether there was an apparent reason to combine the known elements in the fashion claimed" slip op. 14, 82 USPQ2d at 1396.

The Supreme Court further affirmed *In re Kahn,* 441 F. 3d 977, 988 (CA Fed. 2006), stating: "[R]ejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness."

In this regard, it is respectfully submitted that a single use/mention of a system/method by a single reference is insufficient evidence in the record that it would have been obvious to try the same in the primary reference. As relied upon in the *KSR* decision, any underlying obvious to try rationale still requires evidence in the record of the same.

Impermissible hindsight must be avoided and the legal conclusion must be reached on the basis of the facts gleaned from the prior art. See MPEP § 2142 Legal Concept of *Prima Facie* Obviousness. Hindsight cannot be used in determining the issue of obviousness and the

reviewer must view the prior art without reading into that art the teachings of the application or patent (see Kalman v. Kimberly Clark Corp,713 F.2d 760,218 U.S.P.Q. 781(Fed. Cir. 1983)).

In this regard applicants submits to the remainder of the decision particularly indicating that regardless of the rejection rationale, the obviousness reason presented by the Examiner must be more than a mere conclusion, i.e., a brief statement that one skilled in the art would "try" the claimed feature is insufficient. Supported "evidence" for the same is same is still required.

An assessment of basic knowledge and common sense that is not based on any evidence in the record lacks substantial evidence support. Id. at 1385, 59 USPQ 2d at 1697. See also *In re Lee*, 277 F.3d 1338, 1344-45, 61 USPQ 2d 1430, 1434-35 (Fed. Cir. 2002. (In reversing the Board's decision, the court stated "'common knowledge and common sense' on which the Board relied in rejecting Lee's application are not the specialized knowledge and expertise contemplated by the Administrative Procedure Act. Conclusory statements such as those here provided do not fulfill the agency's obligation...The board cannot rely on conclusory statements when dealing with particular combinations of prior art and specific claims, but must set forth the rationale on which it relies."). See MPEP § 2144.03.

B. FIRST GROUND OF REJECTION

In the Office Action, the Examiner rejects claims 1, 3, 6, 13-14, 17, 32 and 35 under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824, and Levy.

An issue is whether Chan in view of combinations of Dearth '242, and Dearth '824, and Levy suggest to one skilled in the art to achieve, the recited features of claims 1, 3, 6, 13-14, 17, 32 and 35.

A first sub-issue is whether the Examiner has established a prima facie case of obviousness based upon Chan in view of combinations of Dearth '242, and Dearth '824, and Levy.

A second sub-issue is whether the Examiner has provided evidence which as a whole show that the legal determination sought to be proved (i.e., whether the reference teachings establish a prima facie case of obviousness) is more probable than not by the preponderance of evidence burden-of-proof standard (37 CFR 1.56(b)).

1) Claim 1

a) First Example of Lack of Support That Recited Features Are Taught By Arguendo Combination of the Art And Lack of Support For Conclusory Comments

Claim 1 recites a method of simulating an operation of a logical unit including "requesting

a resource in which a thread manager, which controls threads each forming an execution unit of a program, makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of a series of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to a resource manager which manages said information about the hardware resource."

The Examiner concedes that Chan does not teach a method requesting a resource in which a thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource, as recited by claim 1. (Action at page 3, lines 11-14).

In support of the rejection the Examiner asserts that Dearth `242, alone, teaches this feature citing Fig. 2, Item 202 and Item 130 and Abstract lines 17-20. (Action at page 3, lines 14-18).

However, in fact, Dearth '242 teaches:

> The barrier mechanism is used to ensure that all tests which are to request reservations of devices of the circuit simulation have requested from the hub such reservations before any test proceeds. In this way, the hub can establish the order in which such requests are granted in a repeatable manner.

(Abstract, lines 17-20)

In Fig. 2, Dearth `242 illustrates that Hub 130 includes a central thread 202 which is the primary thread of hub 130 and which includes a barrier structure 202B.

That is, the Examiner's assertion that Dearth '242 teaches a thread manager that makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource is not supported.

In attempting to support the Examiner's assertion that Dearth '242 teaches requesting a resource in which a thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource the Examiner also asserts:

> [I]it is inherent that when reservations are made and granted, information about hardware resources are requested and sent; the hardware resource information could be the amount of memory allocated or the registers allocated etc.

(Action at page 24, lines 13-15).

Appellants submit that the Examiner has not provided support for such statements of inherency and the statements are merely conclusory and do not support establishment of *prima facie* obviousness.

Appellants submit the rejection of claim 1 is incorrect since none of the art relied on by the Examiner discusses features recited claim 1 as the Examiner asserts and *prima facie* obviousness is not established. Thus, it is submitted that the rejection of claim 1 should be reversed.

b) Second Example of Recited Feature Not Taught By *Arguendo* Combination of the Art And Conclusory Comments Not Supported

The Examiner concedes that neither Chan nor Dearth '242 teach a "allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance," as recited by claim 1 (Action at page 4, lines 8-11).

In attempting to establish *prima facie* obviousness, the Examiner relies on Dearth `824, alone, as teaching "allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance" citing col. 2, lines 46-48. (Action at page 4, lines 11-14).

However, in fact, Dearth '824 merely teaches:

> Without automatic access arbitration, a test design engineer who configures the multiple tests described above would have to include, in the computer instructions and data of each test, implementation of a protocol by which multiple simultaneous attempted interactions with a particular component of the simulated circuit are arbitrated. However, such exacerbates the complexity of the task performed by the test design engineer and is an impractical solution given the typical development process for such tests. In particular, a test design engineer typically designs each test of a simulated circuit individually without consideration of other tests that may be previously or subsequently developed for testing the simulated circuit.

(emphasis added, lines 38-52).

That is, the Examiner's assertion that Dearth `824 teaches allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance is not supported.

The Examiner further asserts:

> Dearth et al. (`242) teaches requesting a resource in which a thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource . . . states that the barrier mechanism is used to ensure that all

> tests which are to request reservations of devices of the circuit simulation have requested from the hub such reservations; the hub can establish the order in which such requests are granted in a repeatable manner; it is inherent that when reservations are made and granted, information about hardware resources are requested, allocated and sent; the hardware resource information could be the amount of memory allocated or the registers allocated etc.). Dearth et al. (' 824), teaches that when simulating a circuit using multiple concurrently executing threads, often more than one thread would attempt to interact with the same simulated component of the simulated circuit or device; and it would be necessary to avoid collisions in multiple concurrently executing threads by reserving simulated devices to one or more simulation threads . . . it is inherent that when reservations are made and granted, information about hardware resources are requested, allocated and sent).

(Emphasis added, Action at page 25, line 10 - page 26, line 7).

Appellants submit that the Examiner has not provided support for such statements of inherency and the statements are merely conclusory and do not support establishment of *prima facie* obviousness.

Appellants submit the rejection of claim 1 is incorrect since none of the art relied on by the Examiner discusses features recited claim 1 as the Examiner asserts and *prima facie* obviousness is not established. Thus, it is submitted that the rejection of claim 1 should be reversed.

c) Third Example of Lack of Support That Recited Features Are Taught By *Arguendo* Combination of the Art

Claim 1 recites a method "controlling a thread in which said thread manager controls an execution state of said thread in accordance with a result of the allocation made by said resource manager, said thread manager and said resource manager executing said requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed simulating the operation of said logical unit to be conducted up to the completion (emphasis added)."

The Examiner concedes that neither Chan nor Dearth `824 teach controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager, the thread manager and the resource manager executing the requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of the thread reaches completion simulating the operation of the logical unit to be conducted up to the completion, as recited by claim 1. (Action at page 5, lines 4-9).

In attempting to establish *prima facie* obviousness, the Examiner relies on Dearth `242, alone, as teaching controlling a thread in which the thread manager controls an execution state of the thread in accordance with a result of the allocation made by the resource manager, the thread manager and the resource manager executing the requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of the thread reaches completion simulating the operation of the logical unit to be conducted up to the completion citing Fig. 2 and Abstract, lines 22-26. (Action at page 5, lines 10-15).

However, in fact, Dearth '242 merely teaches:

> As each test enters the barrier mechanism, execution of the test is suspended and a reference to the test is added to a thread list. When all tests which are to enter the barrier have done so, each thread identified by a reference on the thread list is awakened and execution of the test resumes. simulating the operation of the logical unit to be conducted up to the completion.

(Abstract, lines 22-26)

That is, the Examiners assertion that Dearth '242, alone, teaches the thread manager and the resource manager execute the requesting, allocating, and controlling repeatedly in cooperation with each other is not supported.

Appellants submit the rejection of claim 1 is incorrect since none of the art relied on by the Examiner discusses features recited claim 1 as the Examiner asserts and *prima facie* obviousness is not established. Thus, it is submitted that the rejection of claim 1 should be reversed.

d) Fourth Example of Lack of Support That Recited Features Are Taught By *Arguendo* Combination of the Art

Claim 1 recites a method "controlling a thread in which said thread manager controls an execution state of said thread in accordance with a result of the allocation made by said resource manager, said thread manager and said resource manager executing said requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed simulating the operation of said logical unit to be conducted up to the completion (emphasis added)."

The Examiner concedes that none of Chan, Dearth `242 and Dearth `824 teach dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed,

as recited by claim 1. (Action at page 5, lines 15-18).

In attempting to establish *prima facie* obviousness, the Examiner relies on Levy, alone, as teaching dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the-resource manager every time the generated thread is executed citing col. 3, lines 11-15; col. 3, lines 22-30, and col. 3, lines 35-45. (Action at page 5, lines 18-21).

However, in fact, Levy merely teaches:

> The renaming registers are dynamically allocated and assigned to a thread being processed by the multithreaded processor, as required to support out-of-order execution of instructions for the thread. . . . the renaming registers also comprise sets of renaming registers, each of which is private to a different one of the plurality of threads. In anther preferred embodiment, the renaming registers are dynamically allocatable and assignable to different threads at different times and are thus shared between the threads . . . . The processor determines that an architectural register in a set is temporarily usable as a renaming register by executing an instruction that indicates the architectural register is not currently being used for a thread. In still another preferred embodiment, any of the plurality of registers are dynamically allocatable and assignable to any thread being executed by the multithreaded processor, for use as a renaming register.

(col. 3, lines 10-36).

That is, the Examiners assertion that Levy teaches dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the-resource manager every time the generated thread is not supported.

Appellants submit the rejection of claim 1 is incorrect since none of the art relied on by the Examiner discusses features recited claim 1 as the Examiner asserts and *prima facie* obviousness is not established. Thus, it is submitted that the rejection of claim 1 should be reversed.

e) The Examiner Errs In His Assertion That There Motivation To Combine the Art In the Manner as the Examiner Suggests

Even assuming *arguendo* that the Examiner's assertions regarding recited features are correct, Appellants submit that the Examiner's attempts to combine the art are incorrect. The Examiner asserts it would have been obvious to one of ordinary skill in the art to modify the method of Chan:

> with the method of Dearth et al. (`242) that included requesting a resource in which a thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource because as per Dearth et at. (`824), when simulating a

> circuit using multiple concurrently executing threads, often more than one thread would attempt to interact with the same simulated component of the simulated circuit or device; and it would be necessary to avoid collisions in multiple concurrently executing threads by reserving simulated devices to one or more simulation threads (CL2, L30-33; Abstract, L1-3; CL4, L32-36).

(Emphasis added, Action at page 3, line 18, page 4, line 7).

First, the Examiner is incorrect in asserting that Dearth `824 teaches "when simulating a circuit using multiple concurrently executing threads." Rather, Dearth '824 teaches

> Collisions in access to a simulated device are avoided by reserving to one of two or more hardware simulation tests the simulated device.

(abstract)

and further teaches:

> Without automatic access arbitration, a test design engineer who configures the multiple tests described above would have to include, in the computer instructions and data of each test, implementation of a protocol by which multiple simultaneous attempted interactions with a particular component of the simulated circuit are arbitrated. However, such exacerbates the complexity of the task performed by the test design engineer and is an impractical solution given the typical development process for such tests. In particular, a test design engineer typically designs each test of a simulated circuit individually without consideration of other tests that may be previously or subsequently developed for testing the simulated circuit.

(col. 2, lines 42 54).

That is, Dearth '824 discusses "tests" and not "threads.". Moreover, even assuming the *arguendo* that Dearth '824 "tests" teach "threads", Dearth '824 does not teach avoiding collisions in multiple concurrently executing threads by reserving simulated devices to one or more simulation threads. Rather, Dearth '824 teaches:

> [A] relatively simple, yet effective, access arbitration mechanism in which deadlock and starvation in serving device reservation requests of multiple, concurrently executing tests are prevented and in which arbitration of such multiple reservation requests is completely repeatable to facilitate analysis of tests and circuit simulations.

(col. 4, lines 32-38)

and,

> Deadlocks involving requests of multiple tests for reservation of devices are prevented by establishing the order in which such requests are served and requiring that a test must first relinquish reservation of all devices prior to reserving additional devices. Thus, when the additional requests are appended to a queue of pending reservation requests, no test whose requests follow the requests of a second test in the queue can reserve a device requested by the second test. In other words, the situation in which each of two or more tests has reserved a device, reservation of which is required by another of the two or more tests, cannot occur. Starvation is prevented by combining the sorted queue of each reservation phase into a

> sorted "round robin" arrangement. Specifically, each collection of requests of a reservation phase are sorted into a request arrival queue and, when the request arrival queue includes all requests of a particular reservation phase, the request arrival queue is appended to the pending request queue.

(Abstract)

That is, Dearth '824 does not teach avoiding collisions in multiple concurrently executing threads by reserving simulated devices to one or more simulation threads, but rather by either having only one test run at a time or establishing a queue order. Thus, the Examiner's rationale for modifying Chan with Dearth '242 based on a motivation in Dearth '824 is not supported.

However even assuming that the Examiner's assertions regarding Dearth '824 are correct, Appellants submit there is no motivation to modify Chan with Dearth '242 to avoid collisions of threads. Chan teaches:

> Thus at the beginning of each simulation cycle (step 54 of FIG. 11), the master thread will check if the current time is at the beginning of a clock-cycle boundary. If it is not, it will invoke threads to evaluate event-driven regions only (steps 55-57 of FIG. 11). Otherwise, it will invoke threads to process both event-driven and cycle-based regions. Threads for cycle-based regions will evaluate logic gates in their assigned regions (steps 59-60 of FIG. 11), whereas threads for event-driven regions will do work only if there are events in their event queue for this clock cycle time (steps 55-57). In addition to the above, at the end of each simulation cycle (step 58 of FIG. 11), the master thread will advance the simulation time to either the next event time or clock cycle time, whichever comes sooner.

(col. 12, lines 42-55).

That is, since Chan teaches avoiding collisions of threads through clock cycle management, if Chan is modified in a manner as the Examiner asserts so that the threads which check time and advance simulation time are modified to make request for information about hardware resources, Appellants submit Chan, as illustrated in Chan Fig. 11, would no longer advance the simulation time as designed and be unworkable.

Since as relied upon in the *KSR* decision, any underlying obvious to try rationale still requires evidence in the record of the same, Appellants submit that the *prima facie* obviousness is not established since there is no evidence in the record supporting such a proposed medication of Chan and the rejection of claim 1 should be reversed.

f) Second Example that the Examiner Errs In His Assertion That There is Motivation To Combine the Art In A Manner as the Examiner Suggests

Even assuming *arguendo* that Levy teaches "dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource

manager every time the generated thread is, Appellants submit that the Examiner is incorrect in his contention that:

> it would have been obvious to one of ordinary skill in the art at the time of Applicants' invention to modify the method of Chan, Dearth et at.(`242) and Dearth et al. (`824) with the method of Levy et al. that included dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed because that would support out-of-order execution of instructions for the threads (CL3, L14-15) with dynamic scheduling of instructions thus improving the performance of multithreaded processor (CL1, L38-41).

(Emphasis added, Action at page 6, lines 1-5).

As discussed above, Chan teaches avoiding collisions of threads through clock cycles and if Chan is modified in a manner as the Examiner asserts so that the support out-of-order execution of instructions for the threads with dynamic scheduling of instructions, Appellants submit Chan, as illustrated in Chan Fig. 11, would no longer advance the simulation time and be unworkable.

Since as relied upon in the *KSR* decision, any underlying obvious to try rationale still requires evidence in the record of the same, Appellants submit that the *prima facie* obviousness is not established since there is since there is no evidence in the record supporting such a proposed modification of Chan and the rejection of claim 1 should be reversed.

g) Summary

Since even an *arguendo* combination of the art relied on by the Examiner does not teach recited features of claim 1, it is submitted that *prima facie* case of obviousness is not established. Further, even if the art relied on by the Examiner does teach recited features of claim 1, it is submitted that an *arguendo* combination is unworkable and without motivation for such a modification and a *prima facie* case of obviousness is not established.

Further, the Examiner has not provided evidence showing that the reference teachings establish a *prima facie* case of obviousness as more probable than not by the preponderance of evidence burden-of-proof standard. Thus, the Examiner's rejection of claim 1 is incorrect, and the rejection of claim 1 should be reversed.

2) Claims 13 and 14

The Examiner rejects claim 13 on the same grounds as claim 1. (Action at pages 6-8). The Examiner rejects claim 14 on the same grounds as claim 1 and "based on the same reasoning" as claim 13. (Action at page 8).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 1, Appellants submit that the Examiner makes similar errors in the rejections of claims 13 and 14.

In attempting to establish *prima facie* obviousness, the Examiner relies on Dearth `824, alone, as teaching "a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of the thread; and resource allocating means for allocating information about a hardware resource relating to a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance," as recited by claim 13 citing col. 2, lines 46-48. (Action at page 7).

However, in fact, Dearth '824 merely teaches:

> Without automatic access arbitration, a test design engineer who configures the multiple tests described above would have to include, in the computer instructions and data of each test, implementation of a protocol by which multiple simultaneous attempted interactions with a particular component of the simulated circuit are arbitrated. However, such exacerbates the complexity of the task performed by the test design engineer and is an impractical solution given the typical development process for such tests. In particular, a test design engineer typically designs each test of a simulated circuit individually without consideration of other tests that may be previously or subsequently developed for testing the simulated circuit.

(emphasis added, lines 38-52).

That is, the Examiner's assertion that Dearth `824 teaches allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance is not supported.

The Examiner concedes that neither Chan nor Dearth '824 teach "resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread to the resource manager." (Action at page 7, lines 14-17). The Examiner relies on Dearth '242, alone, as teaching "resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread to the resource manager," as recited by claim 13. citing Fig. 2, Item 202 and Item 130 and Abstract lines 17-20 (Action at page 7, lines 15-19). However, in fact, Dearth '242 teaches:

> The barrier mechanism is used to ensure that all tests which are to request reservations of devices of the circuit simulation have requested from the hub such reservations before any test proceeds. In this way, the hub can establish the order in which such requests are granted in a repeatable manner.

(Abstract, lines 17-20).

In Fig. 2, Dearth `242 illustrates that Hub 130 includes a central thread 202 which is the primary thread of hub 130 and which includes a barrier structure 202B.

That is, the Examiner's assertion that Dearth '242 teaches a thread manager that makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource is not supported.

The Examiner concedes that Chan and Dearth `824 do not teach thread control means for controlling an execution state of the thread in accordance with a result of a resource allocation made by the resource manager in response to the request from the resource requesting means; and the thread manager and the resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other, as recited by claims 13 and 14.

The Examiner relies on Dearth `242 as teaching thread control means for controlling an execution state of the thread in accordance with a result of a resource allocation made by the resource manager in response to the request from the resource requesting means; and the thread manager and the resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other, citing Fig. 2 and Abstract lines 22-26. (Action at page 8).

However, in fact, Dearth '242 merely teaches:

> As each test enters the barrier mechanism, execution of the test is suspended and a reference to the test is added to a thread list. When all tests which are to enter the barrier have done so, each thread identified by a reference on the thread list is awakened and execution of the test resumes. simulating the operation of the logical unit to be conducted up to the completion.

(Abstract, lines 22-26).

That is, the Examiners assertion that Dearth '242, alone, teaches the thread manager and the resource manager execute the requesting, allocating, and controlling repeatedly in cooperation with each other is not supported.

The Examiner concedes that Chan, Dearth `824 and Dearth "242 do not teach dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed. (Action at page 8).

The Examiner relies on Levy as teaching "dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed," as recited by claim 13. (Action at page 8). However, in fact, Levy merely teaches that:

> renaming registers are dynamically allocated. . . any of the plurality of registers are dynamically allocatable and assignable to any thread being executed by the multithreaded processor, for use as a renaming register.

(col. 3, lines 10-36).

That is, the Examiners assertion that Levy teaches dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is not supported.

Summary

Since even an *arguendo* combination of the art relied on by the Examiner does not teach recited features of claims 13 and 14, it is submitted that a *prima facie* case of obviousness is not established. Further, even if the art relied on by the Examiner does teach recited features of claims 13 and 14, it is submitted that an arguendo combination is unworkable and without motivation for such and a *prima facie* case of obviousness is not established.

Further, the Examiner has not provided evidence showing that the reference teachings establish a *prima facie* case of obviousness as more probable than not by the preponderance of evidence burden-of-proof standard. Thus, the Examiner's rejection of claims 13 and 14 is incorrect, and the rejection of claims 13 and 14 should be reversed.

3) Claims 3 and 32

Claim 3 calls for the method simulating an operation of a logical unit according to claim 1, wherein said series of functions are represented in a plurality of sequential or concurrently executed threads. Claim 32 calls for the computer-readable recording medium storing a computer-readable program according to claim 14, wherein said series of functions are represented in a plurality of sequential or concurrently executed threads.

The Examiner rejects claims 3 and 32 on the same grounds as claim 1. (Action at pages 6 and 9). As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 1, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 3 and 32.

That is, the Examiner's assertion that Dearth '242 teaches a thread manager that makes a request for information about a hardware resource relating to a hardware resource needed for

execution of each of threads, to a resource manager which manages the information about the hardware resource is not supported. Further, the Examiners assertion that Dearth '242, alone, teaches the thread manager and the resource manager execute the requesting, allocating, and controlling repeatedly in cooperation with each other is not supported. Further, the Examiners assertion that Levy teaches dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the-resource manager every time the generated thread is not supported.

Summary

Since even an *arguendo* combination of the art relied on by the Examiner does not teach recited features of claims 3 and 32, it is submitted that a *prima facie* case of obviousness is not established. Thus, the Examiner's rejection of claims 3 and 32 is incorrect, and the rejection of claims 3 and 32 should be reversed.

4) Claims 6, 17, and 35

Claim 6 recites a method of simulating an operation of a logical unit according to claim 1, wherein said resource manager monitors resource requests in said requesting a resource to make a decision on a resource request deadlock state among a plurality of threads as a result of the monitoring. Claim 17 recites a computer-readable recording medium storing a computer-readable program according to claim 14, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads as a result of the monitoring. Claim 35 recites a computer-readable recording medium storing a computer-readable program according to claim 32, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads on a result of the monitoring.

The Examiner rejects claims 6, 17, and 35 on the same grounds as claim 1. (Action at pages 6 and 9). The Examiner rejects claims 17 and 35 based on the same reasoning as claim 6. (Action at page 9).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 1, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 6, 17, and 35.

Summary:

Since *prima facie* case of obviousness is not established, the rejection of claims 6, 17, and 35 should be reversed.

C. SECOND GROUND OF REJECTION

1) Claims 2 And 23

Claim 2 recites a method of simulating an operation of a logical unit according to claim 1, wherein said series of functions are represented in a plurality of sequential threads. Claim 23 recites a computer-readable recording medium storing a computer-readable program according to claim 14, wherein said series of functions are represented in a plurality of sequential threads.

The Examiner rejects claims 2 and 23 under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824, Levy, and further in view of Kinzelman.

The Examiner concedes that none of Chan, Dearth `242, Dearth ` 824 and Levy teach that the series of functions are represented in a plurality of sequential threads. However, the Examiner asserts that Kinzelman teaches this feature and it would have been obvious to one of ordinary skill to modify the method of Chan, Dearth `242, Dearth`824 and Levy. with the method of Kinzelman because that would allow instruction threads to be synchronized to align responder instructions from one transactor with the appropriate commander instructions. (Action at pages 9-10).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 1, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 2 and 23.

Further, Appellants submit that if Chan is modified in a manner as the Examiner asserts so that the threads which check time and advance simulation time are modified to allow instruction threads to be synchronized Chan as designed would be unworkable.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 2 and 23 should be reversed.

2) Claim 26

Claim 26 recites a computer-readable recording medium storing a computer-readable program according to claim 23, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads on a result of the monitoring. The Examiner rejects claim 26 on the same basis as claim 23 and 17. (Action at page 10).

As discussed in Section VII. B. 4) pointing out the errors in the Examiner's rejection of claim 17, the Examiner has not provided any motivation or rationale to support such a

modification of Chan, Dearth `242 and Levy to provide such a monitoring of "resource requests in the requesting a resource to make a decision on a resource request deadlock." Accordingly, the Examiner has not supported a finding of *prima facie* obviousness.

Summary:

Since *prima facie* case of obviousness is not established, the rejection of claim 26 should be reversed.

D. Third Ground Of Rejection

Claims 4, 5, 15, 16, 33 and 34 are rejected under 35 U.S.C. 103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824 and Levy and further in view of De Yong.

1) Claim 4

The Examiner concedes that Chan, Dearth `242, and Levy do not teach that a plurality of resource managers each corresponding to the resource manager are provided in conjunction with the types of the information about hardware resources, and in the allocating a resource, each of the resource managers allocates the information about a hardware resource, the resource manager manages, to the thread in accordance with a local rule described in advance, as recited by claim 4. (Action at pages 10-11). The Examiner asserts that this feature is taught by Dearth `824. (Action at page 11).

The Examiner concedes that none of Chan, Dearth `242, Dearth '824 and Levy teach a plurality of resource managers each corresponding to the resource manager are provided in conjunction with the types of the hardware resources, as recited by claim 4.

The Examiner asserts that this feature is taught by De Yong and it would have been obvious to modify the method of Chan, Dearth `242, Dearth ` 824 and Levy with the method of De Yong et al. because a plurality of hierarchical resource managers (arbitration systems) would provide an ordered resolution of temporal contentions. Appellants submit that in fact, Dearth '824 merely teaches:

> Without automatic access arbitration, a test design engineer who configures the multiple tests described above would have to include, in the computer instructions and data of each test, implementation of a protocol by which multiple simultaneous attempted interactions with a particular component of the simulated circuit are arbitrated. However, such exacerbates the complexity of the task performed by the test design engineer and is an impractical solution given the typical development process for such tests. In particular, a test design engineer typically designs each test of a simulated

> circuit individually without consideration of other tests that may be previously or subsequently developed for testing the simulated circuit.

(emphasis added, lines 38-52).

That is, the Examiner's assertion that Dearth `824 teaches allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance is not supported.

Summary

Since *prima facie* obviousness is not established, the rejection of claim 4 should be reversed.

2) Claim 5

The Examiner concedes that Chan, Dearth `242, Dearth `824, and Levy do not teach a plurality of resource managers each corresponding to the resource manager are provided in conjunction with the types of the information about hardware resources and are hierarchized according to the dependence among the information about hardware resources, and in the resource allocating, the allocation of information about a hardware resource is made in consideration of the dependence between the information about a hardware resource managed by one of the resource managers and the information about a hardware resource managed by the other resource manager lower in hierarchy than the one of the resource managers, as recited by dependent claim 5. The Examiner asserts that De Yong et al. teaches citing col. 19, lines 35-36:

> a plurality of resource managers each corresponding to the resource manager are provided in conjunction with the types of the information about hardware resources and are hierarchized according to the dependence among the information about hardware resources, and in the resource allocating, the allocation of information about a hardware resource is made in consideration of the dependence between the information about a hardware resource managed by one of the resource managers and the information about a hardware resource managed by the other resource manager lower in hierarchy than the one of the resource managers.

(Action at page 12).

However, De Yong merely teaches "Hierarchical arbitration systems provide an ordered resolution of temporal contentions." (col. 19, lines 35-36).

De Yong does not teach, for example, that the resource managers are provided in conjunction with the types of the information about hardware resources, as recited by claim 5, as the Examiner asserts.

De Yong does not teach, for example, that the resource managers are hierarchized according to the dependence among the information about hardware resources, and in the resource allocating, the allocation of information about a hardware resource is made in consideration of the dependence between the information about a hardware resource managed by one of the resource managers and the information about a hardware resource managed by the other resource manager lower in hierarchy than the one of the resource managers, as recited by claim 5, as the Examiner asserts.

Further, the Examiners suggested motivation to modify Chan, Dearth `242, Dearth ` 824, and Levy so hierarchical resource managers (arbitration systems) would provide an ordered resolution of temporal contentions does not provide motivation as to why one would modify the art to include information about a hardware resource managed by the other resource manager lower in hierarchy than the one of the resource managers, for example.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claim 5 should be reversed.

3) Claims 15, 16, 33 and 34,

The Examiner rejects claims 15 and 33 based on the same reasoning as claim 4 and rejects claims 16 and 34 based on the same reasoning as claim 5. (Action at page 13). As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 4, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 15 and 33.

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 5, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 16 and 34.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 15, 16, 33 and 34 should be reversed.

E. FOURTH GROUND OF REJECTION

1) Claims 7, 18, and 36

Claims 7 is rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Levy. (Action at pages 13-14).

The Examiner rejects claims 18 and 36 based on the same reasoning as claim 7. (Action at page 14).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claims 1 and 14, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 7, 18, and 36.

Further, the Examiner concedes that neither Chan, Dearth `242, and Levy teach that the resource manager monitors read/write requests with respect to the information about a hardware resource allocated by the resource request in the requesting a resource to make a decision on a competition state in read/write operation on the information about a hardware resource among a plurality of threads on the basis of a result of the monitoring. The Examiner asserts that Dearth '824 teaches this feature. (Action at pages 13-14).

However, the Examiner has not provided any motivation or rationale to support such a modification of Chan, Dearth `242 and Levy to provide such a monitoring of monitors read/write requests with respect to the information about a hardware resource allocated by the resource request in the requesting a resource to make a decision on a competition state in read/write operation on the information."

Accordingly, the Examiner is in error in supporting a finding of *prima facie* obviousness.

Summary:

Since *prima facie* case of obviousness is not established, the rejection of claims 7, 18, and 36 should be reversed.

F. FIFTH GROUND OF REJECTION

Claims 8, 10, 19, 21, 37, and 39 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Markov.

1) Claims 8, 19, and 37

The Action concedes that Chan, Dearth `242, Dearth '824 and Levy do not teach that the resource manager monitors the number of resource requests with respect to the information about a hardware resource to detect a bottleneck on the thread on the basis of a result of the monitoring. Markov teaches that the resource manager monitors-the-number of resource requests with respect to the information about a hardware resource to detect a bottleneck on the thread on the basis of a result of the monitoring, as recited by claim 8. (Action at page 14).

The Examiner rejects claims 19 and 37 based on the same reasoning as claim 8.

(Action at page 15). The Examiner asserts the feature is taught by Markov and it would have been obvious to modify the method of Chan, Dearth `242, Dearth `824 and Levy to detect a bottleneck on the thread on the basis of a result of the monitoring because that would allow the resource manager to intervene and control the bottlenecks and allow evolutionary generation of candidate architectures.

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claim 1, 14, and 32, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 8, 19, and 37.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 8, 19, and 37 should be reversed.

2) Claims 10, 21, and 39

The Action concedes that Chan, Dearth `242, Dearth `824 and Levy do not expressly teach that the thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by the resource manager. The Action asserts the feature is taught by Markov.

However, the Examiner has not provided any motivation or rationale to support such a modification of Chan, Dearth `242 and Levy to provide thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by the resource manager.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 10, 21, and 39 should be reversed.

G. SIXTH GROUND OF REJECTION

1) Claims 9, 20, and 38

Claims 9, 20, and 38 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Markov and Kasuya and rejections based on the same reasoning. (Action at pages 15-16).

The Action concedes that Chan, Dearth `242, Dearth `824 and Levy do not expressly teach that the resource manager monitors the number of resource requests with respect to the information about a hardware resource to detect blocking of the resource requests on the basis

of a result of the monitoring, as recited by claims 9, 20, and 38. The Examiner asserts Kasuya teaches the feature and it would have been obvious to modify the method of Chan, Dearth `242, Dearth ` 824 and Levy with the method of Kasuya.

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claims 1 and 14, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 9, 20, and 38.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 9, 20, and 38 should be reversed.

H. SEVENTH GROUND OF REJECTION

1) Claims 11, 22, and 40

Claims 11, 22, and 40 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Furuichi and rejections based on the same reasoning. .

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claim 1, and 14. Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 8, 19, and 37

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 11, 22, and 40 should be reversed.

I. EIGHTH GROUND OF REJECTION

1) Claims 12 and 41

Claims 12 and 41 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of Dearth '242, Dearth '824 and Levy 175, and further in view of Hollander and rejections based on the same reasoning.

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 1, Appellants submit that the Examiner makes similar errors in the rejections of claims 12 and 41. In attempting to establish *prima facie* obviousness, the Examiner relies on Dearth `824, alone, as teaching "a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of the thread; and resource allocating means for allocating information about a hardware resource relating to a hardware resource meeting the

request to the thread in accordance with a rule prescribed in advance," as recited by claim 13 citing col. 2, lines 46-48. (Action at pages 18-19). However, in fact, Dearth '824 merely teaches:

> Without automatic access arbitration, a test design engineer who configures the multiple tests described above would have to include, in the computer instructions and data of each test, implementation of a protocol by which multiple simultaneous attempted interactions with a particular component of the simulated circuit are arbitrated. However, such exacerbates the complexity of the task performed by the test design engineer and is an impractical solution given the typical development process for such tests. In particular, a test design engineer typically designs each test of a simulated circuit individually without consideration of other tests that may be previously or subsequently developed for testing the simulated circuit.

(emphasis added, lines 38-52).

That is, the Examiner's assertion that Dearth `824 teaches allocating a resource in which the resource manager allocates the information about a hardware resource meeting the request to the thread in accordance with a rule prescribed in advance is not supported.

The Examiner concedes that neither Chan nor Dearth '824 teach "resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread to the resource manager." (Action at page 7, lines 14-17).

However, the Examiner relies on Dearth '242, alone, as teaching "resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread to the resource manager," as recited by claim 13. citing Fig. 2, Item 202 and Item 130 and Abstract lines 17-20 (Action at page 7, lines 15-19).

However, in fact, Dearth '242 teaches:

> The barrier mechanism is used to ensure that all tests which are to request reservations of devices of the circuit simulation have requested from the hub such reservations before any test proceeds. In this way, the hub can establish the order in which such requests are granted in a repeatable manner.

(Abstract, lines 17-20)

In Fig. 2, Dearth `242 illustrates that Hub 130 includes a central thread 202 which is the primary thread of hub 130 and which includes a barrier structure 202B.

That is, the Examiner's assertion that Dearth '242 teaches a thread manager that makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads, to a resource manager which manages the information about the hardware resource is not supported.

The Examiner concedes that Chan and Dearth `824 do not teach thread control means for controlling an execution state of the thread in accordance with a result of a resource allocation made by the resource manager in response to the request from the resource requesting means; and the thread manager and the resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other, as recited by claims 13 and 14.

The Examiner relies on Dearth `242 as teaching thread control means for controlling an execution state of the thread in accordance with a result of a resource allocation made by the resource manager in response to the request from the resource requesting means; and the thread manager and the resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other, citing Fig. 2 and Abstract lines 22-26. (Action at page 19).

However, in fact, Dearth '242 merely teaches:

> As each test enters the barrier mechanism, execution of the test is suspended and a reference to the test is added to a thread list. When all tests which are to enter the barrier have done so, each thread identified by a reference on the thread list is awakened and execution of the test resumes. simulating the operation of the logical unit to be conducted up to the completion.

(Abstract, lines 22-26)

That is, the Examiners assertion that Dearth '242, alone, teaches the thread manager and the resource manager execute the requesting, allocating, and controlling repeatedly in cooperation with each other is not supported.

The Examiner concedes that Chan, Dearth `824 and Dearth '242 do not teach dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed. (Action at page 19).

The Examiner relies on Levy as teaching "dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the resource manager every time the generated thread is executed," as recited by claim 13. (Action at pages 19-20). However, in fact, Levy merely teaches that:

> renaming registers are dynamically allocated. . . any of the plurality of registers are dynamically allocatable and assignable to any thread being executed by the multithreaded processor, for use as a renaming register.

(col. 3, lines 10-36).

That is, the Examiners assertion that Levy teaches dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by the-resource manager every time the generated thread is not supported.

Summary

Since even an *arguendo* combination of the art relied on by the Examiner does not teach recited features of claims 12 and 41, it is submitted that a *prima facie* case of obviousness is not established. Further, even if the art relied on by the Examiner does teach recited features of claims 12 and 41, it is submitted that an arguendo combination is unworkable and without motivation for such and a *prima facie* case of obviousness is not established.

Further, the Examiner has not provided evidence showing that the reference teachings establish a *prima facie* case of obviousness as more probable than not by the preponderance of evidence burden-of-proof standard. Thus, the Examiner's rejection of claim 13 is incorrect, and the rejection of claims 12 and 41 should be reversed.

J. NINTH GROUND OF REJECTION

1) Claims 24-25

Claims 24 and 25 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman and De Yong. (Action at page 21).

The rejection of clam 24 is based on the rejection of claims 15 and 23. The Examiner has not provided any motivation or rationale to support such a modification of Chan, Dearth `242 and Levy to provide such a monitoring of monitors read/write requests with respect to the information about a hardware resource allocated by the resource request in the requesting a resource to make a decision on a competition state in read/write operation on the information." Accordingly, the Examiner is in error in supporting a finding of *prima facie* obviousness.

The rejection of claim 25 is based on the rejection of claims 16 and 25. As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of claim 15, 16, 23, and 25, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 24 and 25.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claims 24 and 25 should be reversed.

K. TENTH GROUND OF REJECTION

1) Claim 27

Claim 27 is rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman. The Examiner's support for the rejection of claim 27 is the same as claim 18. (Action at pages 21-22). However, the Examiner has not provided any motivation or rationale to support such a modification of Chan, Dearth `242 and Levy to provide such a monitoring of monitors read/write requests with respect to the information about a hardware resource allocated by the resource request in the requesting a resource to make a decision on a competition state in read/write operation on the information."

Accordingly, the Examiner is in error in supporting a finding of *prima facie* obviousness.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claim 27 should be reversed.

L. ELEVENTH GROUND OF REJECTION

1) Claims 28 and 30

Claims 28 and 30 are rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and further in view of Kinzelman and Markov. The Examiner's support for the rejection is the same as for claims 8 and 10. (Action at page 22). As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claim 14, Appellants submit that the Examiner makes similar errors in the rejection of dependent claims 28 and 30

Summary

Since *prima facie* case of obviousness is not established, the rejection of claim 28 and 30 should be reversed.

M. TWELFTH GROUND OF REJECTION

1) Claim 29

Claim 29 is rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Kasuya. The Examiner's support for the rejection is the same as for claims 20 and 23. (Action at page 22).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent

claim 1, Appellants submit that the Examiner makes similar errors in the rejection of dependent claim 29 .

Summary

Since *prima facie* case of obviousness is not established, the rejection of claim 29 should be reversed.

N. THIRTEENTH GROUND OF REJECTION

1) Claim 31

Claim 31 is rejected under 35 U.S.C. §103(a) as being unpatentable over Chan in view of combinations of Dearth '242, Dearth '824, and Levy and further in view of Kinzelman, Markov, and Furuichi. The Examiners support for the rejection is the same as for claims 22 and 23. (Action at page 23).

As discussed in Section VII. B. 1) regarding errors in the Examiner's rejection of parent claim 1, 14, and 32, Appellants submit that the Examiner makes similar errors in the rejection of dependent claim 31.

Summary

Since *prima facie* case of obviousness is not established, the rejection of claim 31 should be reversed.

Overall Summary

In view of the law and facts stated herein, the Appellant respectfully submits that the Examiner has failed to establish the obviousness rejection against the pending claims, and the Examiner's findings of unpatentability regarding claims 1-41 should be reversed and the patentability over the presently cited references be affirmed.

The Commissioner is hereby authorized to charge any additional fees required in connection with the filing of this Appeal Brief to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: October 5, 2007

By Paul W. Bobowiec
Paul W. Bobowiec
Registration No. 47,431

1201 New York Ave, N.W., Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

VIII: CLAIMS APPENDIX

1. A method of simulating an operation of a logical unit, comprising:

requesting a resource in which a thread manager, which controls threads each forming an execution unit of a program, makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of a series of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to a resource manager which manages said information about the hardware resource;

allocating a resource in which said resource manager allocates said hardware resource meeting said request to said thread in accordance with a rule prescribed in advance; and

controlling a thread in which said thread manager controls an execution state of said thread in accordance with a result of the allocation made by said resource manager,

said thread manager and said resource manager executing said requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed simulating the operation of said logical unit to be conducted up to the completion.

2. The method of simulating an operation of a logical unit according to claim 1, wherein said series of functions are represented in a plurality of sequential threads.

3. The method of simulating an operation of a logical unit according to claim 1, wherein said series of functions are represented in a plurality of sequential or concurrently executed threads.

4. The method of simulating an operation of a logical unit according to claim 1, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of said hardware resources, and

in said allocating a resource, each of said resource managers allocates said information about a hardware resource, said resource manager manages, to said thread in accordance with a local rule described in advance.

5. The method of simulating an operation of a logical unit according to claim 1, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of said hardware resources and are hierarchized according to the dependence among said information about hardware resources, and

in said resource allocating, the allocation of information about a hardware resource is made in consideration of the dependence between said information about a hardware resource managed by one of said resource managers and said information about a hardware resource managed by the other resource manager lower in hierarchy than the one of said resource managers.

6. The method of simulating an operation of a logical unit according to claim 1, wherein said resource manager monitors resource requests in said requesting a resource to make a decision on a resource request deadlock state among a plurality of threads as a result of the monitoring.

7. The method of simulating an operation of a logical unit according to claim 1, wherein said resource manager monitors read/write requests with respect to said information about a hardware resource allocated by said resource request in said requesting a resource to make a decision on a competition state in read/write operation on said information about a

hardware resource among a plurality of threads on the basis of a result of the monitoring.

8. The method of simulating an operation of a logical unit according to claim 1, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect a bottleneck on said thread on the basis of a result of the monitoring.

9. The method of simulating an operation of a logical unit according to claim 1, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect blocking of said resource requests on the basis of a result of the monitoring.

10. The method of simulating an operation of a logical unit according to claim 1, wherein said thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by said resource manager.

11. The method of simulating an operation of a logical unit according to claim 1, wherein said thread has an execution time-limit on said function.

12. A method of simulating an operation of a logical unit, comprising:

requesting a resource in which a thread manager, which controls threads each forming an execution unit of a program, makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of a series of threads representative of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to a resource manager which manages said information about the hardware resource;

allocating a resource in which said resource manager allocates said information about a hardware resource meeting said request to said thread in accordance with a rule prescribed in advance;

controlling a thread in which said thread manager controls an execution state of said thread in accordance with a result of the allocation made by said resource manager,

with said thread manager and said resource manager executing said requesting, allocating, and controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed, simulating the operation of said logical unit to be conducted up to the completion;

comparing a result of the simulation with an estimated value on said operation of said logical unit; and

outputting a result of the comparison to an external unit.

13. An apparatus for simulating an operation of a logical unit, comprising:

a thread manager for controlling a thread forming an execution unit of a program; and

a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of said thread,

said thread manager including:

resource requesting means for making a request for information about a hardware resource relating to a hardware resource needed for execution of a thread representative of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to said resource manager; and

thread control means for controlling an execution state of said thread in accordance with a result of a resource allocation made by said resource manager in response to

the request from said resource requesting means,

said resource manager including:

resource allocating means for allocating information about a hardware resource relating to a hardware resource meeting the request to said thread in accordance with a rule prescribed in advance,

said thread manager and said resource manager conducting the resource request and the control of the thread execution state repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed, for simulating the operation of said logical unit to be conducted up to the completion.

14. A computer-readable recording medium storing a program for simulation of an operation of a logical unit, the program comprising computer instructions which when executed on a computer makes the computer function as a thread manager for controlling threads each forming an execution unit of said program and as a resource manager for managing information about a hardware resource relating to a hardware resource needed for execution of each of threads, by:

requesting a resource in which said thread manager makes a request for information about a hardware resource relating to a hardware resource needed for execution of threads representative of a series of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to said resource manager;

allocating a resource in which said resource manager allocates said information about a hardware resource meeting the request to said thread in accordance with a rule prescribed in advance; and

controlling a thread in which said thread manager controls an execution state of said

thread in accordance with a result of the allocation made by said resource manager,

said thread manager and said resource manager executing the requesting, the allocating, and the controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed, simulating the operation of said logical unit to be conducted up to the completion.

15. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources, and

in said resource allocating, each of said resource managers allocates said information about a hardware resource, said resource manager manages, to said thread in accordance with a local rule described in advance.

16. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources and are hierarchized according to the dependence among said information about a hardware resource, and

in said resource allocating, the information about a hardware resource allocation is made in consideration of the dependence between said information about a hardware resource managed by one of said resource managers and said information about a hardware resource managed by the other resource manager lower in hierarchy than the one of said resource managers.

17. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads as a result of the monitoring.

18. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said resource manager monitors read/write requests with respect to said information about the hardware resource allocated by said resource request in said resource requesting to make a decision on a competition state in read/write operation on said information about the hardware resource among a plurality of threads on the basis of a result of the monitoring.

19. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said resource manager monitors the number of resource requests with respect to said information about the hardware resource to detect a bottleneck on said thread on the basis of a result of the monitoring.

20. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect blocking of said resource requests on the basis of a result of the monitoring.

21. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by said resource manager.

22. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said thread has an execution time-limit on said function.

23. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said series of functions are represented in a plurality of sequential threads.

24. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources, and

in said resource allocating, each of said resource managers allocates said information about a hardware resource, said resource manager manages, to said thread in accordance with a local rule described in advance.

25. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources and are hierarchized according to the dependence among said information about a hardware resource, and

in said resource allocating, the information about a hardware resource allocation is made in consideration of the dependence between said information about a hardware resource managed by one of said resource managers and said information about a hardware resource managed by the other resource manager lower in hierarchy than the one of said resource managers.

26. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads on a result of the monitoring.

27. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said resource manager monitors read/write requests with respect to said information about a hardware resource allocated by said resource request in said resource requesting to make a decision on a competition state in read/write operation on said information about a hardware resource among a plurality of threads on the basis of a result of the monitoring.

28. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect a bottleneck on said thread on the basis of a result of the monitoring.

29. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect blocking of said resource requests on the basis of a result of the monitoring.

30. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by said resource manager.

31. The computer-readable recording medium storing a computer-readable program according to claim 23, wherein said thread has an execution time-limit on said function.

32. The computer-readable recording medium storing a computer-readable program according to claim 14, wherein said series of functions are represented in a plurality of sequential or concurrently executed threads.

33. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources, and

in said resource allocating, each of said resource managers allocates said information about a hardware resource, said resource manager manages, to said thread in accordance with a local rule described in advance.

34. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein a plurality of resource managers each corresponding to said resource manager are provided in conjunction with the types of information about a hardware resource relating to hardware resources and are hierarchized according to the dependence among said information about a hardware resource, and

in said resource allocating, the information about a hardware resource allocation is made in consideration of the dependence between said information about a hardware resource managed by one of said resource managers and said information about a hardware resource managed by the other resource manager lower in hierarchy than the one of said resource managers.

35. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein said resource manager monitors resource requests in said resource requesting to make a decision on a resource request deadlock state among a plurality of threads on a result of the monitoring.

36. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein said resource manager monitors read/write requests with respect to said information about a hardware resource allocated by said resource request in said resource requesting to make a decision on a competition state in read/write operation on said information about a hardware resource among a plurality of threads on the basis of a result of the monitoring.

37. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect a bottleneck on said thread on the basis of a result of the monitoring.

38. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein said resource manager monitors the number of resource requests with respect to said information about a hardware resource to detect blocking of said resource requests on the basis of a result of the monitoring.

39. The computer-readable recording medium storing a computer-readable program to claim 32, wherein said thread has a budget on a time of occupancy of information about a hardware resource relating to a hardware resource allocated by said resource manager.

40. The computer-readable recording medium storing a computer-readable program according to claim 32, wherein said thread has an execution time-limit on said function.

41. A computer-readable recording medium storing a computer-readable program for simulation of an operation of a logical unit, the program comprising computer instructions which when executed on a computer makes the computer execute:

requesting a resource in which a thread manager, which controls threads each forming an execution unit of a program, makes a request for information about a hardware resource relating to a hardware resource needed for execution of each of threads representative of functions required until the operation of said logical unit reaches completion according to a design specification of said logical unit, to a resource manager which manages said information about a hardware resource;

allocating a resource in which said resource manager allocates said hardware resource meeting said request to said thread in accordance with a rule prescribed in advance;

controlling a thread in which said thread manager controls an execution state of said thread in accordance with a result of the allocation made by said resource manager,

said thread manager and said resource manager executing the requesting, the allocating, and the controlling repeatedly in cooperation with each other until the execution of said thread reaches completion while dynamically allocating information about a hardware resource relating to necessary hardware resources to the thread by said resource manager every time the generated thread is executed, simulating the operation of said logical unit to be conducted up to the completion;

comparing a result of the simulation with an estimated value on said operation of said logical unit; and

outputting a result of the comparison to an external unit.

IX. EVIDENCE APPENDIX

None

X. RELATED PROCEEDINGS APPENDIX

None